

A System for the Animation of Virtual Humans

Nadia Magnenat Thalmann

MIRALab, University of Geneva
University of Geneva
24, rue du Général-Dufour
CH 1211 Geneva, Switzerland
E-mail: thalmann@cui.unige.ch
WWW: <http://miralabwww.unige.ch>

Daniel Thalmann

Computer Graphics Lab
Swiss Federal Institute of Technology (EPFL)
CH 1015 Lausanne, Switzerland
E-mail: thalmann@lig.di.epfl.ch
WWW: <http://ligwww.epfl.ch>

Abstract

Several very complex problems must be solved in order to create true three-dimensional virtual humans in their environment. In this paper, we explain several of these problems and present solutions. In particular, we discuss the creation of virtual humans, the animation of the body and the face, the autonomy of virtual actors, their role in Virtual Reality, the simulation of cloth and hair.

1. Introduction

The ultimate reason for creating virtual humans (also called synthetic actors) who have autonomous behaviors and who seem real is to be able to interact with them in any virtual scene representing the real world. Anyway, a virtual scene, beautiful though it may be, is not complete without people...virtual people that is. Scenes involving virtual humans imply many complex problems that we try to manage since several years [1]. We slowly come to the point of simulating 3D real-looking virtual humans, taking into account body, face and cloth deformations. Any environment could be simulated and consequently, we will be able to experiment in real-time any virtual environment, and to communicate with virtual humans rather naturally. The main problem is this research is to avoid special effects, 2D effects and direct kinematics techniques like keyframe and rotoscoping. We look after procedural methods to simulate the appearance and the functions of life. If our simulation is based on procedural methods, it is then possible to interact with these actors in virtual reality. In this case, our actors have in themselves a description of their look and also know how to behave and respond to stimuli of their virtual environment and also to stimuli or interaction with the real world.

2. Creation of Virtual Humans

The synthesis of realistic virtual humans leads to obtain and include the specific features of the character of interest. For the universally known personalities (actors) such as Marilyn, Humphrey, and Elvis, there is a less scope to make mistakes as the deviations will be very easily detected by the spectator. In spite of this ambition to make realism, or better, imitation, this type of realism should not be confused with the photographic or the cinematographic realism.

Sculpting the shape of a virtual actor

Creating a body for a virtual human is only the first step, his particular character depends on his body movements and his personality is defined by the subtle changes of his facial expressions and other gestures.

To construct these shapes, we propose the use of an interactive sculpting approach. The surfaces of human face and body are irregular structures implemented as polygonal meshes. We have introduced a methodology [2] for interactive sculpting using a six-degree-of-freedom interactive input device called the SpaceBall. When used in conjunction with a common 2D mouse, full three dimensional user interaction is achieved, with the SpaceBall in one hand and the mouse in the other. The SpaceBall device is used to move around the object being sculpted in order to examine it from various points of view, while the mouse carries out the picking and deformation work onto a magnifying image in order to see every small detail in real time (e.g. vertex creation, primitive selection and local surface deformations). In this way, the user not only sees the object from every angle but he can also apply and correct deformations from every angle interactively.

3. Motion Control Methods

Although many techniques have been developed for the control of articulated bodies; they are generally applied to much simpler systems than humans. For a general **locomotor system**, only a combination of various techniques may result in a realistic motion with a relative efficiency. Consequently, our locomotor system is based on several integrated methods. As we have already developed several techniques: keyframe, inverse kinematics, direct/inverse dynamics (Figure 1), biomechanics-based walking, and automatic grasping, we integrate them using a blending approach. Production of a natural looking motion based on the integration of all parts of body using different types of motion control methods is certainly a challenge. In our case, a blending module is associated with the coach-trainee correction method that we have created in the context of walking. This method allows the kinematics correction of joint-space based motion with respect to cartesian constraints [3]. In such a way, it is still possible to modify the key frame sequence, a low-level description of motion, for a higher level goal-oriented requirement. We believe that this approach will greatly extend the scope of predefined motions (motion capture, specialized model, key-framed etc..). A new methodology emerges for motion conception and editing which is centered on the coach-trainee correction method. The functional diagram in Figure 2 organizes the composition of the input motions around the blending module prior to the coach-trainee correction module.

Figure 1. Dynamics-based motion

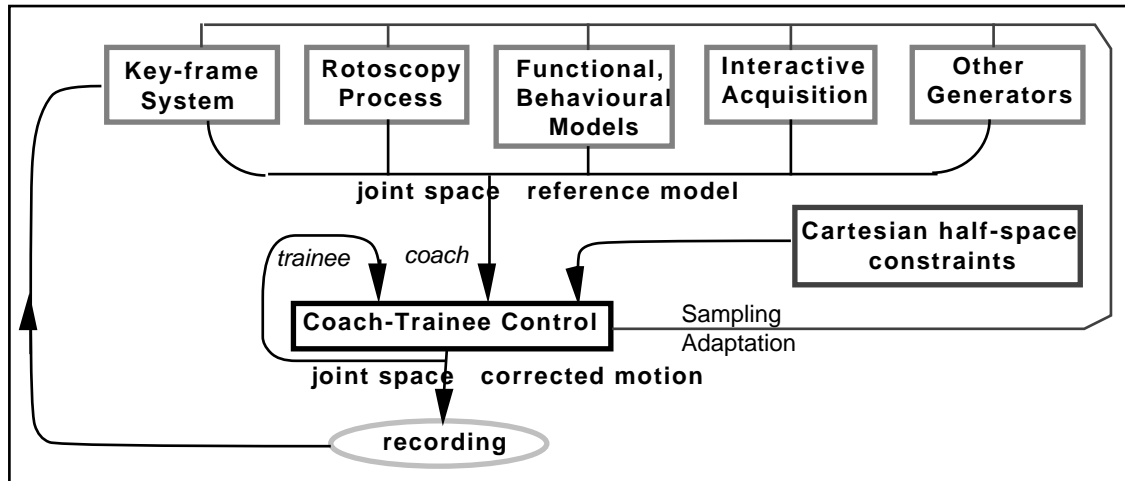


Figure 2. Functional diagram of motion blending

To individualize human walking, we have developed [4] a model built from experimental data based on a wide range of normalized velocities (see Figure 3). The model is structured on two levels. At a first level, global spatial and temporal characteristics (normalized length and step duration) are generated. At the second level, a set of parameterized trajectories produce both the position of the body in space and the internal body configuration, in particular the pelvis and the legs. This is performed for a standard structure and an average configuration of the human body. The experimental context corresponding to the model is extended by allowing continuous variation of the global spatial and temporal parameters for altering the motion to try to achieve the effect desired by the animator. The model is based on a simple kinematics approach designed to preserve the intrinsic dynamic characteristics of the experimental model. But what is important is that this approach allows individualization of the walking action in an interactive real-time context in most cases.

Figure 3. Biomechanics walking

Our grasping approach is based on three steps:

1. **Inverse kinematics to find the final arm posture**
2. **Heuristic grasping decision.**

Based on a grasp taxonomy [5] (see Table 1), the grasping procedure is automatic selected. In particular, the system can decide to use a pinch when the object is too small to be grasped by more than two fingers or to use a two-handed grasp when the object is too large. For a more general

object, the decision should be made according to its multiple bounding volumes. Figure 4 shows examples produced by the system.

Type	Size	Grasping way
Cube	thin	lateral
	thick	pinch
Sphere	small diameter	tripod
	large diameter	wrap
Cylinder and Frustum	small diameter	pinch
	large diameter	wrap
	small height	disk

Table 1. The grasping decision

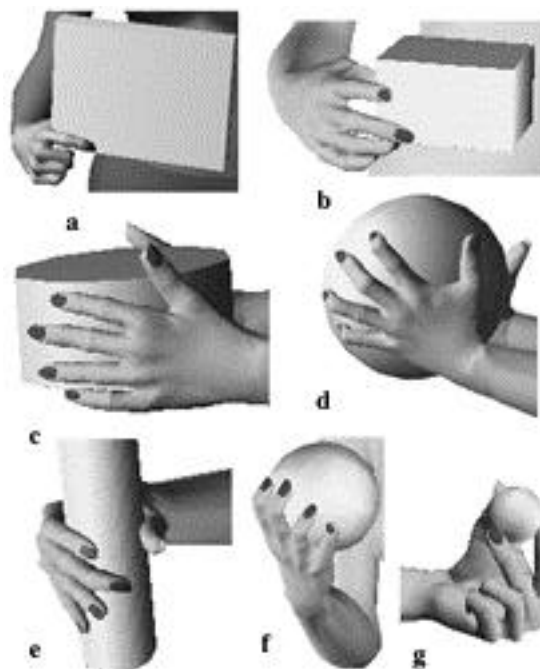


Figure 4. The different grasping ways for different objects a. cube lateral; b. cube pinch; c. cylinder wrap; d. sphere wrap; e. cylinder wrap; f. sphere wrap; g. two hands sphere tripod

3. Multi-sensor hand

Spherical multi-sensors (Figure 5) are attached to the articulated hand. Each sphere sensor is fitted to its associated joint shape with different radii. When a sensor is activated in a finger, only the articulations above it stop moving, while others can still move. By doing this, all the fingers are finally positioned naturally around the object.

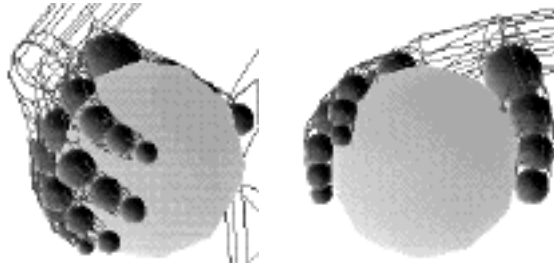


Figure 5. One example shown sensors in grasping

4. Body deformations

Realistic modeling and deformation of human body shapes is an important but difficult problem. We use a multi-layered approach for deforming human bodies [6].

Ellipsoidal metaballs are used to simulate the gross behavior of bone, muscle, and fat tissue; they are attached to the skeleton and arranged in an anatomically-based approximation. The skin construction is made in a three step process. First, the implicit surface resulting from the combination of the metaballs influence is automatically sampled along cross-sections with a ray casting method. Second, the sampled points constitute control points of a B-spline patch for each body part (limbs, trunk, pelvis, neck). Third, a polygonal surface representation is constructed by tessellating those B-spline patches for seamless joining different skin pieces together and final rendering.

Figure 6 shows an example

Figure 6. Example of human body

In our metaball editor, we start the shape design by first creating a articulated skeleton for the organic body to be modeled. Metaballs are used to approximate the shape of internal structures which have observable effect on the surface form. Each metaball is attached to its proximal joint, defined in the joint local coordinate system of the underlying skeleton which offers a convenient reference frame for position and editing metaball primitives, because the relative proportion, orientation, and size of different body parts are already well-defined.

Supporting the five normalized scaling is straight forward in our model, because we simply need to scale the axis lengths of 3 principal directions of each metaball accordingly. Figure 7 illustrates the effect of the scaling operators on the skin envelope. Moreover, the interpolation of the metaballs parameters between two key designs can generate interesting 3D morphing of human shape.

Figure 7. Automatic skin scaling.

5. Facial Animation

It is difficult to create a model for facial animation that is physically realistic confirming to the anatomical details of facial topology while convenient for animators to use and manipulate. We use a multi-level approach [7], which divides the facial animation problem into a hierarchy of levels that are independent of each other. The higher levels use higher degree of abstraction for defining the entities such as emotions and phrases of speech, by those facilitating animators to manipulate these entities in a natural and intuitive way. From the highest to the lowest, the levels of abstraction encompassed in our system are illustrated in Figure 8. The high level layers are the most abstract and specify "what to do", the low level layers describe "how to do". Each level is seen as an independent layer with its own input and output.

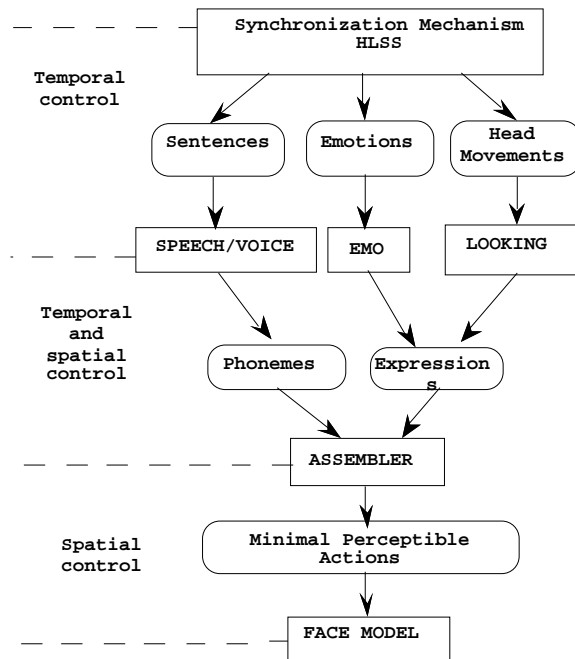


Figure 8. Hierarchical Structure of the Multi-level Facial Animation System

A synchronization mechanism is provided at the top level that requires animators to specify the highest level entities: emotions, sentences and head movement with their durations. The system provides default values for durations in case they are not specified. These entities are then decomposed into lower level entities and sent through the pipeline of control at lower levels of the hierarchy. The temporal characteristics of animation are generally controlled at higher levels and the spatial characteristics are controlled at lower levels in our multi level system

For our facial deformations, we have extended the concept of Free Form Deformations (FFD). Physically, FFD corresponds to deformations applied to an imaginary parallelepiped of clear, flexible plastic in which are embedded the objects to be deformed. The objects are also considered to be flexible so that they are deformed along with the plastic that surrounds them. A grid of control points is imposed on the parallelepiped. For any point interior to the parallelepiped, the deformation is specified by moving the control point(s) from their undisplaced latticial position. As an extension to the basic FFDs, we provide the option of including rational basis functions in the formulation of deformation [8]. These rational basis functions allow incorporation of weights defined for each of the control points in the parallelepiped grid. The advantage of using rational FFDs is that they provide one more degree of freedom of manipulating the deformations by changing the weights at the control points.

Region build-up is accomplished by interactive selection of polygons. These regions on the human face generally correspond to their anatomical descriptions, such as nose, lips, eyes, etc. The magnitude and direction of the muscle pulled is interactively adjusted by changing the position and weight of the control points on the control-unit of a region. Certain points in a defined region can be set "anchored" meaning they will not deform when the deformations are applied on the region. The physical characteristics of facial mesh points can also be set interactively.

Expressions and phonemes in our system are considered as facial snapshots, that is, a particular position of the face at a given time. For phonemes, only the mouth portion (lips) is considered during the emission of sound. A facial snapshot consists of one or more minimal perceptible actions (MPAs) with their intensity specified. A minimal perceptible action is a basic facial motion parameter. Each MPA has a corresponding set of visible features such as movement of eyebrows, movement of jaw, or mouth and others which occur as a result of contracting and pulling of muscles associated. The set of MPAs included is general enough to account for most possible facial expressions. A generic expression can be represented as follows:

```
[expression <name>
  [mpa <name-1> intensity <i-1>]
  [mpa <name-1> intensity <i-1>]
  ...
]
```

e.g.

```
[expression surprise
  [mpa openjaw intensity 0.17]
  [mpa puffcheeks intensity -0.41]
  [mpa stretch_cornerlips intensity -0.50]
  [mpa raise_eyebrows intensity 0.40]
  [mpa close_lower_eyelids intensity -0.54]
  [mpa close_upper_eyelids intensity -0.20]
]

phoneme ee
  [mpa stretch_cornerlips intensity -0.29]
  [mpa openjaw intensity 0.50]
  [mpa lower_cornerlips intensity -0.57]
  [mpa raise_upperlips intensity -0.20]
]
```

We need a mechanism of synchronization to ensure smooth flow of emotions and sentences with head movements. A language HLSS (High Level Script Scheduler) is used to specify the synchronization in terms of an action and its duration. From the action dependence the starting time and the terminating time of an action can be deduced. The general format of specifying an action is as follows: while <duration> do <action>.

Figure 9 shows an example.

Figure 9. Facial expression

6. Autonomous Virtual Humans

Perception through Virtual Sensors

For VR and multimedia applications, digital actors should be able to react to their environment and take action based on perception. The problem of simulating the behavior of a synthetic actor in an environment may be divided into two parts: 1) provide to the actor a knowledge of his

environment, and 2) to make react him to this environment. A typical simulation is produced in a synchronous way by a behavioral loop such as:

```
t_global = 0.0
    code to initialize the animation environment
while (t_global < t_final) {
    code to update the scene
    for each actor
        code to realize the perception of the environment
        code to select actions based on sensorial input, actual state and specific behavior
    for each actor
        code executing the above selected actions
    t_global += t_interval
}
```

The first problem consists of creating an information flow from the environment to the actor. This synthetic environment is made of 3D geometric shapes. One solution is to give the actor access to the exact position of each object in the complete environment database corresponding to the synthetic world. This solution could work for a very "small world", but it becomes impracticable when the number of objects increases. Moreover, this approach does not correspond to reality where people do not have knowledge about the complete environment. A better solution is to equip the actors with virtual sensors: visual, auditory, and tactile.

Virtual vision

We first introduced [9] the concept of synthetic vision as a main information channel between the environment and the virtual actor. Each pixel of the vision input has the semantic information giving the object projected on this pixel, and numerical information giving the distance to this object. The synthetic actor perceives his environment from a small window of typically 30x30 pixels in which the environment is rendered from his point of view. As he can access z buffer values of the pixels, the color of the pixels and his own position he can locate visible objects in his 3D environment. This information is sufficient for some local navigation.

Virtual audition

In real life, the behavior of persons or animals is very often influenced by sounds. For this reason, we developed a framework [10] for modeling a 3D acoustic environment with sound sources and microphones. The acoustic environment is composed of sound sources and a propagation medium. The sound sources can produce sound events composed of a position in the world, a type of sound, and a start and an end time of the sound. The propagation medium corresponds to the sound event handler which controls the sound events and transmits the sounds to the ears of the actors and/or to a user and/or a soundtrack file.

Virtual tactile

At basic level, human should sense physical objects if any part of the body touches them and gather sensory information. This sensory information is made use of in such tasks as reaching out for an object, navigation etc. In our work [11], we use sphere multi-sensors for tactile perception and grasping as explained in Section 3.

Perception-based actions

Synthetic vision, audition and tactile allow the actor to perceive the environment. Based on this information, his behavioral mechanism will determine the actions he will perform. Actions may be at several degrees of complexity. An actor may simply evolve in his environment or he may interact with this environment or even communicate with other actors. We will emphasize three types of actions: navigation, grasping and tennis.

7. Virtual Humans in Virtual Reality

The VLNET system

One of the most challenging Virtual Reality applications are real-time simulations and interactions with autonomous actors especially in the area of games and cooperative work.

The VLNET [12 13] (Virtual Life NETwork) system supports a networked shared virtual environment that allows multiple users to interact with each other and their surrounding in real time. The users are represented by 3D virtual human actors, which serve as agents to interact with the environment and other agents. The agents have similar appearance and behaviors with the real humans, to support the sense of presence of the users in the environment. In addition to user-guided agents, the environment can also be extended to include fully autonomous human agents used as a friendly user interface to different services such as navigation. Virtual humans can also be used in order to represent the currently unavailable partners, allowing asynchronous cooperation between distant partners. The environment incorporates different media; namely sound, 3D models, facial interaction among the users, images represented by textures mapped on 3D objects, and real-time movies. The environment works as a general stream to include these different media.

Participant, user-guided and autonomous real-time creatures

As virtual actors play a key role in VLNET, we will first try to clarify the concept of virtual actor. We define a virtual (or synthetic) actor as a human-like entity with an abstract representation in the computer. A real-time virtual actor is a virtual actor able to act at the same speed as a real person. Virtual Reality, Interactive Television, and Games require real-time virtual actors. In VLNET, three types of real-time virtual actors may coexist in the same shared environment.

A **participant actor** is a virtual copy of the real user or participant. His movement is exactly the same as the real user. This can be best achieved by using a large number of sensors to track every degree of freedom in the real body, however this is generally not possible due to limitations in number and technology of the sensing devices. Therefore, the tracked information is connected with behavioral human animation knowledge and different motion generators in order to "interpolate" the joints of the body which are not tracked.

A **guided actor** is an actor completely controlled in real-time by the user. In VLNET, the best example of actor guidance is guided navigation. More details on the control of guided actors are given in next section.

As already explained in details in the previous sections, an **autonomous actor** is an actor who may act without intervention of the user. Including autonomous actors that interact with participants increases the real-time interaction with the environment, therefore we believe that it contributes to the sense of presence in the environment. The autonomous actors are connected to the VLNET system in the same way as human participants, and also enhance the usability of the environment by providing services such as replacing missing partners, providing services such as helping in navigation.

Guided Actor Control

For the current implementation, we use the behaviors of walking for navigation and grasping for picking. We also include a set of gestures for representing different intentions. This set of behaviors can easily be extended, however these behaviors are sufficient to perform everyday activities, providing minimum set of behaviors to attend virtual meetings.

For navigation, the guidance is achieved as follows. The real person uses the input devices to update the transformation of the eye position of the virtual actor. This local control is used by computing the incremental change in the eye position, and using the walking motor to update the

joint angles. The walking motor is based on the Humanoid walking model, guided by the user interactively or automatically generated by a trajectory. The joint values are set for walking, based on the mathematical parameterization coming from biomechanical experimental data.

For grasping behavior, we make use of the inverse kinematics within the right arm of the virtual actor. The real person uses the input devices to update the position and orientation of the virtual actor's hand. Based on this input, the realistic posture of the right arm is computed within the joint limit constraints. We use simple chain of the right arm for inverse kinematics for real-time motion of the virtual actor. Although we could apply a physically correct method, our concern is more on the visual appearance of the grasping motion.

Applications

With VLNET, it is possible to have players who may be participants, guided or autonomous. For example, a participant could play chess against another participant or an autonomous actor. A 3D puzzle may be solved by two autonomous actors or a participant with the help of a guided actor. VLNET brings together four essential technologies: networked Computer Supported Cooperative Work (CSCW), Virtual Reality, Artificial Life, and Computer Animation.

Implementation of the VLNET system

The network overhead can have a significant effect, especially with increasing number of users. Therefore, it is important to provide low-latency high-throughput connections. Therefore we are experimenting our system over the ATM pilot network, provided to the Swiss Federal Institute of Technology and University of Geneva, by Swiss Telecom. The simulation part should also be performed efficiently using appropriate mechanism.

We include the facial interaction by texture mapping the image containing the user's face on the virtual actor's head. To obtain this, the subset of the image that contains the user's face is selected from the captured image and is sent to other users. To capture this subset of image, we apply the following method: initially the background image is stored without the user. Then, during the session, video stream images are analyzed, and the difference between the background image and the current image is used to determine the bounding box of the face in the image. This part of the image is sent to the other users after optional compression and the receiving side recognizes automatically the type of incoming images.

8. Simulating Autonomous Clothes Using Physics-Based Models

Introduction

Until now, cloth simulation programs have tried to get realistic results for garments worn by an animated body. The garments were considered as a set of flat polygons made of a regular triangle mesh, that were held together by seaming forces. For calculation speed, geometrical assumptions were made to reduce collision detection time according to the specific situation where an actor wears a garment. That situation is in fact a rather "simple" mechanical context with well-defined collision situations, and quite smooth deformation.

Our system [14] intends to get out of this specificity and to be able to simulate cloth in any situation that may be encountered in real life. This involves for example a cloth being put on a body, thrown away, fold, crumpled,... Situations such as crumpling involve very difficult geometrical and mechanical situations, such as numerous and interdependent collisions and very high deformations. Our system should be robust enough to cope with all situations that may occur.

More than just a simulation program that animates cloth worn by moving body, our system is able to simulate a very wide range of deformable surfaces in whatever scene involving moving objects.

Description of the system

A cloth is represented by an assembly of fabric panels, like real-life garments. After 2D design that may be based on real cloth models, these panels are seamed together using mechanical simulation. The panels are then assembled and the garment becomes a single-piece object, that can be manipulated in any way.

The cloth system is divided into two separate parts:

- The 2D panel editor
- The 3D simulation system

The design process goes through three steps:

Garment panel design

Using the 2D panel editor, the panels composing a cloth are defined geometrically. The editor provides help for geometry and dimensioning with different tools, such as grid matching, duplication, symmetries... Seaming lines are then defined between and within the panels. The designer specifies which border lines should be seamed together. The tool provides help for maintaining consistency between the different seamings. The garment is then stored in a file containing panel and seaming definitions.

Garment assembly

Using the 3D simulation system, the panels composing a garment are then seamed together for composing a cloth. The user defines the context in which the garment has to be assembled. Either it may be assembled "in the air", either around a body with several other cloth for getting a dressed character. The panels are interactively moved to an initial position that will ensure correct assembly. Seamings are represented by "elastics" that will match the seaming lines together. Mechanical computation is performed using the forces exerted by the elastics. Once the seaming lines brought together, the different panels are merged together and a single cloth object is obtained, that may be handled as any other object. At this point, the resulting cloth object may be saved for future use, but the mechanical simulation may also proceed directly for computing the animation.

Garment animation

The garment is put into the animated environment, composed by the moving body or any other object composing the scene. The mechanical calculation moves the deformable surface according to its mechanical parameters (elasticity, thickness,...) and the external solicitations:

- Collision response (repulsion and friction) to itself (self-collision)
- Collision response to other animated or fixed objects (other cloth, the body, any other object)
- External force fields (gravity, wind,...)
- Other solicitations (fixed elements, user defined "elastics",...)

The user may interactively change mechanical settings, block some points of the deformable surfaces, or interactively add "elastics" between any object, modifying the animation.

The mechanical model

Models based on global minimization or Lagrangian dynamics formulations are not suited for highly nonlinear situations where discontinuities generated by collisions are numerous. The main idea of our model is to integrate Newton's motion equation $F = ma$ in a direct way to keep quickly evaluated time steps small. Thus, nonlinear behaviors and discontinuous responses such as collisions will be handled in an accurate way. Furthermore, this direct formulation allows us easy

and precise inclusion of any nonlinear mechanical behavior. With such model, we can also act directly on the position and speed of the elements, and thus avoid handling collisions through strong repulsion forces that perturbate the simulation.

The animated deformable object is represented as a particle system by sets of vertices forming heterogeneous triangles, thus allowing surfaces of any shape to be easily modeled and simulated. The object is considered to be isotropic and of constant thickness. Elastic properties of the object are partially described by the standard parameters that are:

- E_0 the Young modulus
- ν the Poisson coefficient
- ρ the density
- T the thickness

To avoid textile to behave like a rubber sheet, non linearity is added to textile's response through Young Modulus.

Using Newton's second law $F=ma$, the motion equation consists of a pair of coupled first-order differential equation, for position and velocity. The system of equations is resolved using second order (midpoint method) of the Euler-Cromer method.

The constraints implied in deformable object motion are subdivided in two categories:

- Continuous constraints, including internal and some external ones, such as wind (represented as a force proportional to the relative velocity of a surface and a viscous fluid) and gravity. Additional external forces are also applied by elastics binding two vertices together. Such elastics allow seaming and interactive manipulation of the objects.
- Discontinuous constraints resulting from collisions with other objects. They induce instantaneous change in the state of the object.

Internal strains are either in-plane, from planar extension and shearing, or out-of-plane, from bending and twisting. In-plane and out-of-plane deformations are evaluated separately. Considering the irregularity of the triangle mesh, the force evaluation should be independent from the size and shape of the triangles.

Collision detection and handling

For dealing with complex collision situations such as crumpling, we need efficient collision and self-collision detection, as well as a robust collision handling. Collision and particularly self-collision detection is often the bottleneck of simulation applications in terms of calculation time, because of the scene complexity that involves a huge number of geometrical tests for determining which elements are colliding.

In our case, the problem is complicated further because we are handling discretized surfaces that may contain thousands of polygons. We also are considering general situations where we cannot make any hypotheses about region proximities. Finally, we have to efficiently detect self-collisions within the surfaces. This prevents the use of standard bounding box algorithms because potentially colliding regions of a surface are always touching each other by adjacency.

A very efficient algorithm [15] for handling this situation has been implemented in the system. This algorithm is based on hierarchisation and takes advantage of the precited adjacency which, combined with a surface curvature criteria, let us skip large regular regions from the self-collision detection. Figure 10 shows an example of animation sequence.

9. Hair Modeling, Animation and Rendering

In the field of human simulation, hair presents one of the most challenging problems. The difficulties of processing hair result from the large number and detailed geometries of the individual hairs, the complex interaction of light and shadow among the hairs, and the small scale of one hair's width in comparison to the rendered image. In fact, there are basically four problems to solve in order to produce realistic animated synthetic actors with hair: hair modeling and creation, hair motion, collision detection and hair rendering.

Hair modeling

The purpose of the hair modeler is to produce a hair segment file for the animation module. To create the hair style, the program offers some interactive facilities. First, the three-dimensional curved cylinder is composed of straight cylindrical segments connected by points. The "in-between" points can be moved in the space to be adjusted, modified, deleted, or added. Just one type of individual hair can be assigned to each triangle in the scalp mesh. However the same hair can be assigned to various triangles. In order to adjust the hair orientation on the curved scalp surface, its direction can be modified by rotating the hair around the normal of its triangle.

After defining each hair format and applying it to the respective triangle, the final style can be defined. The same hair style can have different lengths, by making it grow or shrink using a multiplication factor. A hair style generally has between 100,000 and 150,000 hairs, but this density can be regulated either for individual triangles or the entire scalp. Initially all the hairs placed in the triangle have the same length, orientation and symmetrical position. To make hair styles look more natural, all these parameters can be changed interactively. A random length, orientation and position can be assigned to each triangle hair set.

STYLER maintains a database of information that parameterizes hair growth on each polygon. Each polygon has the following parameters:

- an identifier referring to a 3D curve
- a material identifier for hairs on this polygon
- jitter for the base location of the curve
- orientation of the hairs, called a tangent vector in the world coordinates
- random angle variation applied to the tangent vector
- scaling value applied to the curve
- random value applied to the curve
- density (number of hairs per unit area)
- a seed for the random generator

Hair animation

To generate natural hair animation, physical simulation must be applied. However, precise simulation including collision response is impractical because of the large number of individual hairs. Simplified collision detection method using cylindrical representation has been described by Kurihara et al. (16) and integrated into our system (17).

Hair rendering

Rendering an image of hair with our system [18] involves several steps:

- creating a database of hair segments
- creating shadow buffers from all lights
- rendering the hairless objects using all shadow buffers
- composing the hair on the hairless image

In the hair style rendering module, the process is step by step. First, the shadow of the scene is calculated for each light source i , as well as for the light sources for the hair shadows. The hair shadows are calculated for the object surface and individually for each hair. Finally the hair style is blended into the scene, using all shadow buffers. The result is an image with a three-dimensional realistic hair style rendering where complex shadow interaction and highlight effects can be seen and appreciated. In more detail, the full rendering pipeline may be summarized as follows:

- We take the scene model description and project it onto each light source, creating one scene shadow buffer for each light source.
- We take the hair model and project it onto each light source, creating one hair shadow buffer for each light source. This is done by drawing each hair segment into a Z-buffer based frame buffer and extracting the resulting depth map.
- We compose the depth maps for the scene shadow buffer and the hair shadow buffer, resulting in a single composite shadow buffer for each light source.
- We generate the scene image and its Z-buffer, using the scene model description and the composite shadow buffers as input to the scene renderer, resulting in a fully rendered scene with hair shadows, but no hair.

We blend the hair segments into the scene image, using the scene's Z-buffer to determine visibility and the composite shadow buffers to determine shadowing, yielding the final image with hair and full shadows. For this blending process, each strand of the hair model is breaking into straight 3D line segments. The intensity H of each of the segment's endpoints is determined using a hair intensity equation. shows an example of a synthetic actress with a hairstyle.. shows a hairstyle. Figure 11 shows a synthetic actress with hairstyle.

Figure 11. Synthetic actress with hairstyle

References

- 1 Magnenat-Thalmann N and Thalmann D (1991), "Complex Models for Visualizing Synthetic Actors". IEEE Computer Graphics and Applications, September (1991).
- 2 Paouri A, Magnenat Thalmann N, Thalmann D (1991) Creating Realistic Three-Dimensional Human Shape Characters for Computer-Generated Films, Proc. Computer Animation '91, Springer-Verlag, Tokyo, pp.89-100

- 3 Boulic R, Thalmann D (1992) Combined Direct and Inverse Kinematic Control for Articulated Figures Motion Editing, *Computer Graphics Forum*, Vol. 2, No.4, October 1992, pp.189-202.
- 4 Boulic R, Magnenat-Thalmann N, Thalmann D (1990) A Global Human Walking Model with real time Kinematic Personification, *The Visual Computer*, Vol.6, No6, pp.344-358.
- 5 Mas S.R., Thalmann D. (1994) "A Hand Control and Automatic Grasping System for Synthetic Actors", *Proc. Eurographic'94*, pp.167-178.
- 6 Jianhua S, Thalmann D, "Interactive Shape Design Using Metaballs and Splines", *Implicit Surface '95, Eurographics Workshop on Implicit Surfaces, Grenoble, France, 1995*
- 7 Kalra P, Mangili A, Magnenat Thalmann N and Thalmann D (1991), "SMILE: A Multilayered Facial Animation System" *Proc. IFPI WG 5.10, Tokyo, Japan* (ed Kunii Tosiyasu L) pp. 189-198.
- 8 Kalra P, A. Mangili, N. Magnenat Thalmann, D. Thalmann (1992) Simulation of Facial Muscle Actions Based on Rational Free Form Deformations, *Proc. Eurographics '92, Cambridge*, pp. 59-69.
- 9 Renault O., Magnenat Thalmann N., Thalmann D. (1990) "A Vision-based Approach to Behavioural Animation", *The Journal of Visualization and Computer Animation*, Vol 1, No 1, pp 18-21.
- 10 Noser H., Thalmann D. (1995) "Synthetic Vision and Audition for Digital Actors", *Proc. Eurographics '95*.
- 11 Huang Z., Boulic R., Magnenat Thalmann N., Thalmann D. (1995) "A Multi-sensor Approach for Grasping and 3D Interaction", *Proc. CGI '95*
- 12 Pandzic I., Capin T., Magnenat Thalmann N., Thalmann D., "VLNET: A Networked Multimedia 3D Environment with Virtual Humans", *Proc. Multi-Media Modeling MMM '95, World Scientific, Singapore*
- 13 Thalmann D., Capin T., Magnenat-Thalmann N., Pandzic I.S., "Participant, User-guided, and Autonomous Actors in the Virtual Life Network VLNET", *Proc. ICAT/VRST '95, Chiba, Japan, November 1995*, pp.3-11.
- 14 Volino P., Courchesne M., Magnenat Thalmann N., "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects", *Computer Graphics*, 29, 1995, pp 137-144.
- 15 P. Volino, N. Magnenat Thalmann, "Efficient Self-Collision Detection on Smoothly Discretised Surface Animations using Geometrical Shape Regularity", *Computer Graphics Forum, Proc.Eurographics '94, 13(3), 1994*, pp 155-166.
- 16 Kurihara T., Anjyo K., Thalmann D. (1993) "Hair Animation with Collision Detection", in: *Models and Techniques in Computer Animation, Springer-Verlag, Tokyo*, pp.128-138.
- 17 Daldegan A., Magnenat Thalmann N., Kurihara T., Thalmann D. (1993) "An Integrated System for Modeling, Animating and Rendering Hair", *Proc. Eurographics '93, Computer Graphics Forum, Vol.12, No3*, pp.211-221.
- 18 LeBlanc A., Turner R., Thalmann D. (1991b) "Rendering Hair using Pixel Blending and Shadow Buffers", *Journal of Visualization and Computer Animation* 2, 3, pp. 92-97